Department of Software Technology

Logic Formulation and Introductory Programing (**CCPROG1**)
Machine Project Specification
1st Trimester, SY 2024 – 2025

Deadline: November 25, 2024 (Monday) 7:59 AM via AnimoSpace

Tides of Manila: A Merchant Quest

A turn-based strategy game[1]

Introduction[2]

Welcome to "Tides of Manila: A Merchant's Quest", a text-based strategy game set in the bustling city of Manila during the pre-colonial era. As a young merchant, you've inherited a small trading vessel and a modest amount of gold coins from your late father. You aim to navigate the Pasig River and trade goods with various ethnic groups to accumulate wealth and prestige.

"Balsas de cocos de la Laguna navegando por el rio Pasig." 1857 Painting by Jose Honorato Lozano.

The game takes place in a dynamic and ever-changing environment, where the tides of the river and the volatile prices of goods can greatly impact your success. You'll need to make strategic decisions about which goods to trade and when to take risks.

As you embark on this journey, you'll encounter historical figures, such as Rajah Sulayman and Rajah Lakandula, who will offer rewards and challenges. You'll also face natural disasters which will test your skills and adaptability.

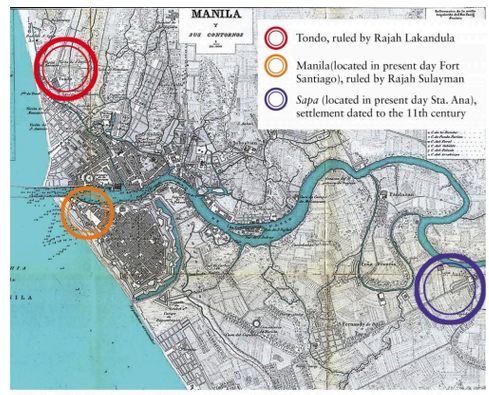
Your ultimate goal is to become the most successful merchant in Manila with the most gold. Will you rise to the challenge and become a legendary merchant, or will you succumb to the dangers and uncertainties of the river?

[1] Inspiration for the game Taipan! by Art Canfil
https://en.wikipedia.org/wiki/Taipan!#Gameplay

[2] from the article https://filipiknow.net/facts-about-pasig-river/

Rules of the Game [3]

1. Trading Ports/Stations



Trading Ports are settlements along the Pasig River that offer trading opportunities for buying and selling goods. These are the following:

- Tondo
- Manila
- Pandakan
- Sapa (present day Sta. Ana)

2. Goods and Prices

For simplicity, goods to be traded are limited to the following:

- Coconut
- Rice
- Silk
- Guns

As you inherited your father's trading vessel, you also received a worn leather-bound book containing the *family's market study*. This ancient tome passed down through generations of merchants, includes valuable information on the price ranges of various goods in each trading location along the Pasig River. The book contains the following information.

Location	Goods	Price Range (in gold coins)
Tondo	Coconut	from 4 to 24
	Rice	from 1 to 20
	Silk	from 48 to 68
	Gun	from 70 to 95
Manila	Coconut	from 3 to 18
	Rice	from 5 to 20
	Silk	from 24 to 39
	Gun	from 65 to 84
Pandakan	Coconut	from 2 to 12
	Rice	from 4 to 14
	Silk	from 22 to 32

[3] Image from https://manilaweekly.wordpress.com/2020/09/13/trivia-no-19-pasig-river-the-life-and-blood-of-manila/

	Gun	from 90 to 103
Sapa	Coconut	from 9 to 14
	Rice	from 1 to 6
	Silk	from 17 to 22
	Gun	from 204 to 301

3. Vessel Cargo Capacity

   Load up to **75 items**, regardless of type of goods, into your vessel, as it has a **maximum capacity** that limits the total number of goods you can transport.

4. Buying and selling goods

   Each time you arrive at a trading post, the market prices of goods are randomly set based on the price range detailed in the family market study (see section above).

   - Rules on Buying Goods
     - When you purchase an item, the corresponding price is multiplied by the quantity, and the total amount is **deducted** from your gold coin balance.
     - Your gold coin balance determines the items you can purchase, as you can only acquire items that fit within the amount of gold coins you currently possess.
     - Purchased items are automatically loaded into the vessel subject to space availability. Thus, the vessel's cargo capacity also limits your purchasing power.
   - Rules on Selling Goods
     - When you sell an item, the corresponding price is multiplied by the quantity, and the total amount is **added** to your gold coin balance.
     - Your vessel cargo determines the items you sell, as you can sell what you have in your cargo.
     - Sold items are automatically unloaded from your vessel.

Game Play

1. Start of Game

   At the start of the game the player is prompted to enter the following information:

   - A Player / Merchant's Code (A positive integer less than or equal to 999)
   - The initial capital (gold coins). This must be a numeric integer greater than zero (0)
   - The target profit percentage. This must be at least twenty (20%)

   Also, take note of the following:

   - game starts with zero (0) cargo on board the vessel

- current *Day* is initially set to zero (0)
- the starting point is the port of Manila

2. The Main navigation screen

After entering the initial information, the control goes to the main navigation screen.

At the **minimum**, the screen should show the following information:

- The Player code
- The Current Location / Trading Post
- Gold coins
- Current Profit (Amount and percentage)
- Cargo Information
- Market Price of Goods
- Current Day

There should also be a list of options for the user to:

- Buy
- Sell
- Go to another port
- Quit

Below is a sample screen for reference:

```
Main Screen

Player:  007                                  Gold Coins: 150
Location: Manila                              Profit      : 0/0%
Day 0

  Cargo              0 of 75
 ====================
  Coconut - 0    Silk - 0
  Rice      - 0  Gun - 0

  Market Prices
 ====================
  Coconut - 3    Silk - 24
  Rice      - 5  Gun - 65

What would you like to do?
    [1] Buy
    [2] Sell
    [3] Go to Another Port

    [Q] Quit
```

3. The Buy Screen

- The *Buy screen* allows the player to purchase an item from a list based on the current market price.
- The player is then asked to enter the qty of goods to buy
- The corresponding cost is then deducted from the player's gold coins
- The purchased quantity is loaded into the vessel
- There should also be a message signifying the status of the transaction (i.e, successful or unsuccessful purchase)
- Control returns to the main screen after

Below is a sample screen for reference:

```
Purchase Screen

Player:  007                                  Gold Coins: 150
Location: Manila                              Profit        : 0/0%
Day 0

  Cargo                0 of 75
  ====================
  Coconut - 0    Silk - 0
  Rice      - 0    Gun - 0

  Market Prices
  ====================
  Coconut - 3    Silk - 24
  Rice      - 5    Gun - 65

What would you like to Buy?
    [1] Coconut
    [2] Rice
    [3] Silk
    [4] Gun

    [X] Return to Main Screen
```

4. <u>The Sell Screen</u>
    - Similar layout to the purchase screen
    - The *Sell screen* allows the player to sell an item from a list based on the current market price.
    - The player is then asked to enter the qty of goods to sell
    - The corresponding cost is then added to the player's gold coins
    - The Sold qty is unloaded from the vessel
    - There should also be a message signifying the status of the transaction (i.e, successful or unsuccessful sale)
    - Control returns to the main screen after

5. <u>The Navigation Screen</u>
    - The navigation screen allows the player to go to another port.
    - Each turn corresponds to 1 Day

- The game turn limit is **30** turns
- There should also be a message signifying the status of the travel (i.e, successful or unsuccessful)
- Control returns to the main screen after travel

Below is a sample screen for reference:

```
Navigation Screen

Player:  007                                  Gold Coins: 150
Location: Manila                              Profit        : 0/0%
Day 0


Where would you like to go?
    [1] Tondo
    [2] Pandakan
    [3] Sapa


    [X] Return to Main Screen
```

NOTE: The current location should NOT be in the list of options.

6. End of Game

The game ends when:

- The Player Quits
- the maximum number of turns (30) is reached.
- the player achieves/surpasses its target profit.

  For instance, if the initial capital is 100 gold coins and the target profit is 30%, the player wins if the gold coins reach or exceed 130 gold coins (100 + 30%) within the 30-turn limit.

An appropriate message should be displayed based on the player's performance.

Below is a sample screen for reference:

```
You Win!

Total Gold 130
Profit 30
Number of turns 15
```

Machine Project Objective and Technical Requirements

For this Machine Project, the student will create a version of this game from scratch using the C Programming language, following the specifications and guidelines detailed in this document.

It is expected that the program you will be developing will require the use of the following concepts:

- Conditions (or Branching Statements)
- Loops (or Iterative Statements)
- User-defined Functions

For random numbers, you may refer to this site:
https://www.geeksforgeeks.org/generating-random-number-range-c/

How to Approach the Machine Project

Step 1: Problem analysis and algorithm formulation

Read the MP Specifications again! Identify clearly what information is required from the user, what processes are needed, and what your program's output (s) will be. Clarify with your professor any issues that you might have regarding the machine project.

When you have all the necessary information, identify the necessary functions that you will need to modularize the project. Identify the required data of these functions and what kind of data they will return to the caller. Write your algorithm for each of these modules/functions and the algorithm for your main program.

Step 2: Implementation

In this step, you must translate your algorithm into proper C statements. While implementing, you are to perform the other phases of program planning and design (discussed in the other steps below) together with this step.

Follow the coding standard indicated in the course notes (Modules section in AnimoSpace).

You may type your program in a text editor or an IDE (i.e. Dev-C IDE) at this point. Note that you are expected to use statements taught in class. You can explore other libraries and functions in C if you clearly explain how these work. You may also use arrays, should these be applicable, and you can properly justify and explain your implementation using these. For topics not covered, it is left to the student to read ahead, research, and explore by himself.

Note, though, that you are NOT ALLOWED to do the following:

- to declare and use global variables (i.e., variables declared outside any function),
- to use goto statements (i.e., to jump from code segments to code segments),
- to use the break or continue statement to exit a block. Break statement can only be used to break away from the switch block,
- to use the return statement or exit statement to prematurely terminate a loop, function, or program,
- to use the exit statement to prematurely terminate a loop or to terminate the function or program and
- call the main() function to repeat the process instead of loops.

It is best that you perform your coding "incrementally." This means:

- Dividing the program specification into subproblems, and solving each problem separately according to your algorithm;
- Code the solutions to the subproblems one at a time. Once you're done coding the solution for one subproblem, apply testing and debugging.

Documentation

While coding, you have to include internal documentation in your programs. You are expected to have the following:

- File comments or Introductory comments
- Function comments
- In-line comments

Introductory comments are found at the very beginning of your program before the preprocessor directives. Follow the format shown below. Note that items in between < > should be replaced with the proper information.

```
/*
     Description:      <Describe what this program does briefly>
     Programmed by:    <your name here>   <section>
     Last modified:    <date when last revision was made>
     Version:          <version number>
    [Acknowledgements: <list of sites or borrowed libraries and sources>]
*/
<Preprocessor directives>

<function implementation>

int main()
{
   return 0;
}
```

Function comments precede the function header. These describe what the function does, the intentions of each parameter, and what is being returned, if any. If applicable, include pre-conditions as well. Pre-conditions refer to the assumed state of the parameters. Follow the format below when writing function comments:

```
/*    <Description of function>
      Precondition: <precondition / assumption>
      @param <name> <purpose>
      @return <description of returned result>
*/
<return type>
<function name> (<parameter list>)
      :
```

Example:

```
/* This function computes for the area of a triangle
   Precondition: base and height are non-negative values
   @param base is the base measurement of the triangle in cm
   @param height is the height measurement of the triangle in
cm
   @return the resulting area of the triangle
*/
float
```

```
getAreaTri (float base,
            float height)
{
    ...
}
```

In-line comments are ot**her comments in major parts of the code. These are expected to explain the purpose or algorithm of groups of related code, esp. for long functions.**

STEP 3: TESTING AND DEBUGGING

Submit the list of test cases you have used. You have to test each feature of your program fully before moving to the next feature. Sample questions that you should ask yourself are:

1. What should be displayed on the screen if the user inputs an order?
2. What would happen if I input incorrect inputs? (e.g., values not within the range)
3. Is my program displaying the correct output?
4. Is my program following the correct sequence of events (correct program flow)?
5. Is my program terminating (ending/exiting) correctly? Does it exit when I press the command to quit? Does it exit when the program's goal has been met? Is there an infinite loop?
7. and others…

SUGGESTED MILESTONES: [INCLUDES CODING, DOCUMENTATION, AND TESTING]

Here is a suggested set of tasks to help you complete all the requirements. Note that you are not expected to submit milestone requirements. All requirements in these specs are due on November 25, 2024 (7:59 AM).

Milestone 1: Target due October 21, 2024

1. Draft outline of functions.
2. Implement initialization functions
3. Implement random price generator function
4. All implemented code must include proper comments

Milestone 2: Target due November 4, 2024

1. Screen flow scaffolding of the entire program
2. Implement Buy and Sell functions
3. Documentation and test scripts of implemented functions

Milestone 3: Target due November 18, 2024

1. Finalize screen information and message display
2. Implement Game Exit function
3. Update documentation and test cases

MP Deadline: November 25, 2024 (M) 7:59 AM

The entire program according to the specification.

IMPORTANT POINTS TO REMEMBER:

1. You are required to implement the project using the C language (C99 and NOT C++). Make sure you know how to compile and run in both the IDE (DEV-C++) and the via the command prompt:

   `gcc –Wall <yourMP.c> -o <yourExe.exe>`

2. The implementation will require you to:
   - Create and Use Functions

     Note: Non-use of self-defined functions will merit a grade of **0** for the **machine project**. Too few self-defined functions may merit deductions. A general rule is to create a separate function for each option described above, unless some features are too similar that one function can serve the purpose for two [or more] of the options. Note that functions whose tasks are only to display are not included in the count for creating user-defined functions.
   - Appropriately use conditional statements, loops and other constructs discussed in class (Do not use brute force solution. **You are not allowed to use goto label statements, exit statements. You are required to pass parameters to functions and not allowed to declare global or static variables.**)
   - Consistently employ coding conventions
   - Include internal documentation (i.e., comments)

3. The deadline for the project is **7:59 AM on November 25, 2024 (Monday)** via submission through **AnimoSpace**. Late submission up to a max of 3 days will incur deductions. That is, submissions from 8:00 AM of November 25 to 7:59 AM of November 26 will incur **10%** deduction, submissions from 8:00 AM of November 26 to 7:59 AM of November 27 will incur **additional 20%** deduction (*total of 30% deduction*), submissions from 8:00 AM of November 27 to 7:59 AM of November 28 will incur additional **additional 30%** deduction (*total of 60% deduction in MP score*), and submissions from 8:00 AM of November 28 will not be accepted anymore as facility is locked. Once locked, no MP will be accepted anymore, resulting in a **0.0** for your machine project.

4. The following are the deliverables:

Checklist:
❒ Upload in AnimoSpace by clicking **Submit Assignment** on Machine Project and adding the following files: ❒ source code* ❒ test script**
❒ email the softcopies of everything as attachments to **YOUR own email address** on or before the deadline

Legend:

***Source Code also includes the internal documentation. The **first few lines of the source code** should have the following declaration (in the comment) **BEFORE** the introductory comment:

```
/*********************************************************************
This is to certify that this project is my own work, based on my personal efforts in studying
and applying the concepts learned. I have constructed the functions and their respective
algorithms and corresponding code by myself. The program was run, tested, and debugged by
my own efforts. I further certify that I have not copied in part or whole or otherwise
plagiarized the work of other students and/or persons.
                                        <your full name>, DLSU ID# <number>
*********************************************************************/
```

Test Script should be in a table format, with the header as shown below. There should be **at least 3 distinct test classes (as indicated in the description) **per function**. There is no need to create test scripts for functions that only perform displaying on screen.

Function Name	#	Test Description	Sample Input (either from the user or to the function)	Expected Result	Actual Result	P/F
getAreaTri	1	**base and height measurements are less than 1.**	**base = 0.25 height = 0.75**	**…**	**…**	
	2	:::				
	3					

Test descriptions are supposed to be unique and should indicate classes/groups of test cases on what is being tested. Given the function getAreaTri(), the following are 3 distinct classes of tests:

i.) testing with base and height values smaller than 1
ii.) testing with whole number values for base and height
iii.) testing with floating point number values for base and height, larger than 1.

The following test descriptions are incorrectly formed:

Too specific: testing with a base containing 0.25 and height containing 0.75
Too general: testing if function can generate correct area of triangle
Not necessary -- since already defined in pre-condition: testing with base or height containing negative values

5. **MP Demo:** You will demonstrate your project on a specified schedule during the last weeks of classes. Being unable to show up on time during the demo or being unable to answer the questions convincingly during the demo will merit a grade of **0.0** for the **MP**. The project is initially evaluated via black box testing (i.e., based on output of the running program). Thus, if the program does not compile successfully using gcc -Wall and execute in the command prompt, a grade of 0 for the project will be incurred. However, a fully working project does not ensure a perfect grade, as the implementation (i.e., correctness and compliance in code) is still checked.
6. Any requirement not fully implemented and instruction not followed will merit deductions.
7. This is an **individual project**. Working in collaboration, asking other people's help, and/or copying other people's work are considered cheating. Cheating is punishable by a grade of **0.0** for CCPROG1 course, aside from which, a cheating case may be filed with the Discipline Office.
8. The above description of the program is the basic requirement. A maximum of 10 points will be given as a bonus. Use of colors may not necessarily incur bonus points. Sample additional features could be:
   (a) Add a feature that allows upgrade/purchase of vessels with a bigger cargo capacity
   (b) Introduce the concept of warehousing to take advantage of the market price fluctuations.

   Note that any additional feature not stated here may be added but **should not conflict with whatever instruction was given in the project specifications**. Bonus points are given upon the discretion of the teacher, based on the difficulty and applicability of the feature to the program. Note that **bonus points can only be credited if all the basic requirements are fully met** (i.e., complete and no bugs).

HONESTY POLICY AND INTELLECTUAL PROPERTY RIGHTS

Honesty policy applies. Please take note that you are NOT allowed to borrow and/or copy-and-paste – in full or in part any existing related program code from the internet or other sources (such as printed materials like books, or source codes by other people that are not online). **You should develop your own codes from scratch by yourself.**